4. Breath First Search

Breath first search (BFS) เป็นขั้นตอนวิธีในการ traverse และ search กราฟเช่นเดียวกับ DFS แต่ BFS ต้องการท่องกราฟให้กระจายแนวกว้างมากที่สุด ดังนั้นผลการจากการทำงานของ BFS จะให้ระยะทางระหว่างโหนดหรือ cell เริ่มต้นกับตนเอง รูปที่ 6 แสดงระยะทางระหว่าง cell เริ่มต้นที่แถว 0 และคอลัมน์ 0 กับ cell อื่น ๆ สำหรับกรณีเดินได้ 4 ทิศ

	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5

รูปที่ 6

การเขียนโปรแกรมสำหรับ BFS นั้นเหมือนกับ DFS แบบ iterative แทบทุกประการ ยกเว้นเลือกใช้ queue แทน stack โปรแกรม 2.6 แสดงการทำงานของ BFS ที่แสดงผลระยะทางระหว่าง cell เริ่มต้นที่แถว 0 และคอลัมน์ 0 กับ cell อื่น

โปรแกรมที่ 2.6 การทำงานของขั้นตอนวิธีของ BFS บนอาร์เรย์ 2 มิติ

```
1.   #include<iostream>
2.   #include<queue>
3.   #define MAX 1005
4.   using namespace std;
5.   // Globals
6.   int M;                  // number of rows
7.   int N;                  // number of columns
8.   bool visit[MAX][MAX];   // keeps tracks of visited cells
9.   int dist[MAX][MAX];
10.  queue< pair<int,int> > Q; // for iterative method
11.  // Forwards
12.  void iterativeBFS(int r, int c);
13.  bool isValid(int r, int c);
14.
15.  int main(){
16.      cin >> M >> N;
17.      cout << "Iterative BFS\n";
18.      dist[0][0] = 0;
19.      iterativeBFS(0,0);
20.
21.
```

```
22.       for(int i=0; i<M; i++){
23.           for(int j=0; j<N; j++)
24.               cout << dist[i][j] << " ";
25.           cout << "\n";
26.       }
27.       cout << endl;
28.       return 0;
29.   }
30.   void iterativeBFS(int r, int c){
31.       // push just visited cell on queue
32.       // mark cell as visited
33.       Q.push({r,c});
34.       visit[r][c] = true;
35.
36.       // iterate until stack is empty
37.       while(!Q.empty()){
38.           pair<int,int> cell = Q.front();
39.           int x = cell.first;
40.           int y = cell.second;
41.           Q.pop();
42.
43.           // order of priority
44.           // up, right, down, left
45.           int dx[] = {-1,0,1,0};
46.           int dy[] = {0,1,0,-1};
47.           for(int i=0; i<4; i++){
48.               int xi = x + dx[i];
49.               int yi = y + dy[i];
50.               if(isValid(xi, yi)){
51.                   Q.push({xi,yi});
52.                   visit[xi][yi] = true;
53.                   dist[xi][yi] = dist[x][y] + 1;
54.               }
55.           }
56.       }
57.   }
58.   bool isValid(int r, int c){
59.       return (r >= 0 && r < M
60.               && c >=0 && c < N
61.               && !visit[r][c]);
62.   }
```

ผลลัพธ์

3 4

```
Iterative BFS
0 1 2 3
1 2 3 4
2 3 4 5
```